DECISION TREE

A **Decision Tree** in machine learning is a popular and intuitive model used for both **classification** and **regression** tasks.

It resembles a flowchart-like structure where decisions are made based on feature values.

The goal is to split the dataset into smaller subsets that make it easier to predict the target variable.

Components of a Decision Tree:

1. **Root Node**: This is the topmost node that represents the entire dataset. It is split based on a feature that best separates the data.
2. **Decision Nodes**: These are the internal nodes where the data is split further based on different features.
3. **Leaf Nodes**: These represent the final classification or output. No further splitting occurs at this point.

Example: Decision Tree for Classifying Pass/Fail Based on Study Hours

Imagine we have a simple dataset with two features: **hours studied** and whether the student has passed or failed an exam.

Hours Studied	Passed?
1	Fail
2	Fail
3	Fail
4	Pass
5	Pass
6	Pass

We want to build a decision tree that can predict whether a student will pass or fail based on the hours studied.

1. Root Node:

The decision tree will start by choosing the best feature to split the data. Here, "Hours Studied" is the feature.

2. Splitting:

Based on the hours studied, the decision tree splits the data at a certain threshold. Let's say it finds that the best split is at **3.5 hours**.

- If **Hours Studied ≤ 3.5**, then it predicts **Fail**.
- If **Hours Studied > 3.5**, then it predicts **Pass**.

3. Tree Representation:

Here is how the decision tree would look:

```scss
        [Root Node]
      Hours Studied <= 3.5?
       /                  \
  Yes (Fail)          No (Pass)
```

 If a student studied 2 hours, the tree checks whether 2 is less than or equal to 3.5. Since the answer is "Yes," it predicts "Fail."

 If a student studied 5 hours, the tree checks whether 5 is greater than 3.5. Since the answer is "Yes," it predicts "Pass."

Key Points:

- **Feature Selection**: Decision trees select features to split the data based on measures like **Gini Impurity** or **Information Gain** (derived from entropy), which help determine which split improves classification the most.
- **Interpretability**: Decision trees are easy to interpret and visualize, making them a great tool for explaining model decisions.
- **Overfitting**: A common challenge with decision trees is overfitting, where the tree becomes too complex and captures noise in the data. Techniques like **pruning** are used to limit the tree's depth.

Conclusion:

A decision tree simplifies complex decision-making by breaking it down into a series of simpler questions. It's a powerful tool in machine learning, especially when interpretability is important.

DECISION TREE CLASSIFIER PROGRAM

Evaluate a **Decision Tree Classifier** model using the famous **Iris dataset**, which contains information about different iris flower species.

Here's an overview of what each part of the program does:

1. Importing Necessary Libraries

- **pandas, numpy**: For data manipulation and numerical operations.
- **matplotlib.pyplot, seaborn**: For data visualization.
- **sklearn.preprocessing.LabelEncoder**: For encoding categorical labels into numeric form.
- **sklearn.model_selection.train_test_split**: To split the data into training and testing sets.
- **sklearn.tree.DecisionTreeClassifier**: The machine learning model used for classification.
- **sklearn.metrics.classification_report, confusion_matrix**: To evaluate the model's performance.

- **sklearn.tree.plot_tree**: To visualize the decision tree.

2. Loading and Exploring the Iris Dataset

```python
df = sns.load_dataset('iris')
```

- The Iris dataset is loaded using seaborn's built-in dataset. It contains 150 samples of iris flowers categorized into three species (setosa, versicolor, virginica), along with 4 attributes (sepal length, sepal width, petal length, and petal width).

```python
df.to_csv('iris.csv')
df.head()
df.info()
df.shape
df.isnull().any()
```

- The dataset is saved as a CSV file (iris.csv), and the head, info, shape, and missing values are checked.

3. Data Visualization

- **Pair Plot**: Visualizes relationships between different features.

```python
sns.pairplot(data=df, hue = 'species')
```

- **Heatmap**: Visualizes the correlation matrix between numerical features.

```python
sns.heatmap(df.corr())
```

4. Preparing Data for Model

- **Target**: The species column is separated as the target variable.

```python
target = df['species']
```

- **Features**: The dataset (minus the target column) is prepared as features.

```python
df1 = df.drop('species', axis=1)
X = df1
```

- **Label Encoding**: Converts the target labels (species names) into numeric form.

```python
le = LabelEncoder()
target = le.fit_transform(target)
y = target
```

5. Train-Test Split

- The dataset is split into training and testing sets in an 80:20 ratio.

X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, random_state = 42)

6. Training the Decision Tree Classifier

- A decision tree classifier is created and trained using the training data.

```python
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

7. Evaluating the Model

- **Classification Report**: Prints precision, recall, f1-score, and accuracy for each class.

```python
y_pred = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test, y_pred))
```

- **Confusion Matrix**: Displays the confusion matrix to visualize the model's performance.

```python
cf_matrix = confusion_matrix(y_test, y_pred)
print(cf_matrix)
```

8. Visualizing the Decision Tree

- The decision tree is plotted using plot_tree().

```python
plt.figure(figsize=(10,10))
plot_tree(decision_tree=dtree, feature_names=df1.columns,
          class_names=["setosa", "vercicolor", "verginica"],
          filled=True, precision=4, rounded=True)
```

Conclusion

This program successfully builds a decision tree model to classify iris species based on petal and sepal measurements. It also evaluates the model's accuracy and visualizes both the relationships between features and the structure of the decision tree.